



IST STREP Project



Deliverable D3.3.1u
Middleware User's Guide
*Multi-Radio Device Management
Layer*

<http://www.ist-plastic.org>



Project Number	:	IST-26955
Project Title	:	PLASTIC
Deliverable Type	:	Report

Deliverable Number	:	D3.3
Title of Deliverable	:	Middleware User's Guide: Multi-Radio Device Management Layer
Nature of Deliverable	:	Report
Dissemination level	:	PUBLIC
Internal Document Number	:	D3.3.1uV0.2
Contractual Delivery Date	:	6/3/2008
Actual Delivery Date	:	6/3/2008
Contributing WPs	:	WP3
Editor(s)	:	Lee Rong
Author(s)	:	Lee Rong
Reviewer(s)	:	

Abstract

The Multi-Radio Device Management layer forms the foundation of the Multi-radio Communication middleware and provides the upper layer with an implementation of the B3G Network Abstraction. It designs the high level abstract classes for each network types that are then substituted by concrete classes, which provide the actual communication with various network interfaces on different types of devices (e.g., PDAs and laptops). Since most of the functionalities provided by these classes are strongly dependent on the hardware, they are defined as abstract classes and must be further implemented to be compliant with the specific targeted device.

The Multi-radio device management layer manages the low-level characteristics of the perceived networks in terms of functionalities and QoS properties. That is, it is in charge of: (i) sensing the available networks and retrieving their characteristics (attributes and offered services), (2) monitoring their status and, (iii) accessing them to exploit the offered services.

This document gives an overview of the PLASTIC Multi-radio Device Management layer implementation addressing PDAs equipped with Windows Mobile. It provides middleware developers with information on how to use the existing modules.

Keyword list

Document History

Version	Type of change	Author(s)
V0.2	First Draft	Lee Rong, Mauro Caporuscio

Document Review

Date	Version	Reviewer	Comment
<date review>	of <version >	<name affiliation> &	<proofed, found weaknesses, corrections ...>

Table of Contents

1	Component Overview	4
2	Deployment	10
2.1	System requirements	10
2.2	Download.....	10
2.3	Configure.....	11
2.4	Compile and deploy	11
3	Tutorial.....	12

1 Component Overview

B3G networks combine multiple wireless networking technologies in order to benefit from their respective advantages and specificities. Further, the increase in computing and communication capacities of portable devices, as well as their mass marketing, makes possible the widespread deployment of such multi-networks pervasive environments, where B3G-capable devices hold several radio interfaces (e.g., UMTS, WiFi, Bluetooth), and the possibility to switch from one radio interface to another. PLASTIC-enabled devices shall then benefit from such a pervasive network by increasing the perimeter of reachable service providers. However, this should not be realized at the expense of a greater complexity. In fact, dealing with B3G networks opens new challenges and issues in the development and deployment of distributed systems. The PLASTIC middleware shall cope with the complexity induced by the heterogeneity of the wireless technologies by hiding it to the user and, to be effective, to the application developers as well. In this context, the *PLASTIC Multi-radio Device Management* exploits B3G Network Abstraction in the PLASTIC conceptual model [PCD1.2] by capturing the various networks and observing their status (e.g., connectivity and quality of service). Furthermore, the *PLASTIC Multi-radio Device Management* abstracts the network properties (e.g., accessibility and offered capabilities) exploiting their diversity (see [PCD3.3]).

This document gives a detailed description of the *PLASTIC Multi-radio Device Management* layer and provides middleware developers with information on how to use the existing modules.

Provider

INRIA

Introduction

The Multi-Radio Device Management layer manages the low-level characteristics of the perceived networks in terms of functionalities and QoS properties. That is, it is in charge of:

- a. Sensing the available networks and retrieving their characteristics (attributes and offered services),
- b. Monitoring their status proactively or reactively,
- c. Accessing them to exploit the offered services.

As shown in Figure 1, the Multi-Radio Device Management layer forms the foundation of the PLASTIC Communication Middleware. The Multi-Radio Networking layer and the B3G Communication layer rest on the top to form a three-tier architecture. The Multi-Radio Device Management layer can be utilized by the upper layers (e.g., Multi-Radio Network uses Multi-Radio Device Management to switch between different types of networks during network selection), or exploited directly by the application layer.

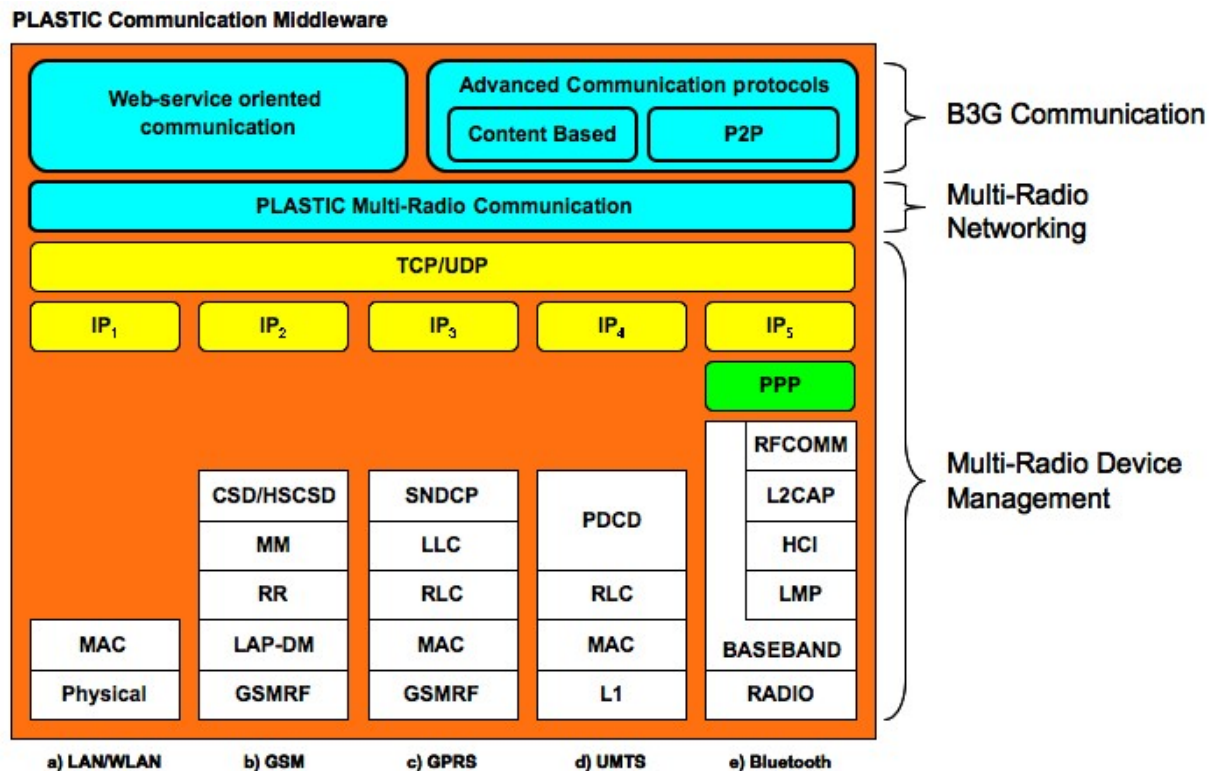


Figure 1. PLASTIC Communication Middleware

This document aims to provide application developers with information on how to use the existing code of the Multi-radio Device Management layer for further development. The Multi-radio Device Management module is divided into two parts: a set of Java based classes to provide API based access to network radio adapters, and a set of drivers which interacts with different hardware components of mobile devices (see Figure 2). Currently, a set of drivers has been implemented in C# for communicating with Windows Mobile based PDAs. The C# code communicates with the Java classes using a local socket connection (e.g., the Java code calls methods in the C# code by sending it messages using the local socket connection). Also, there is a set of dummy drivers for testing purposes. The rest of the document describes the required hardware components and applications for development of the Multi-radio Device Management module, as well as its loading, compilation and deployment procedures.

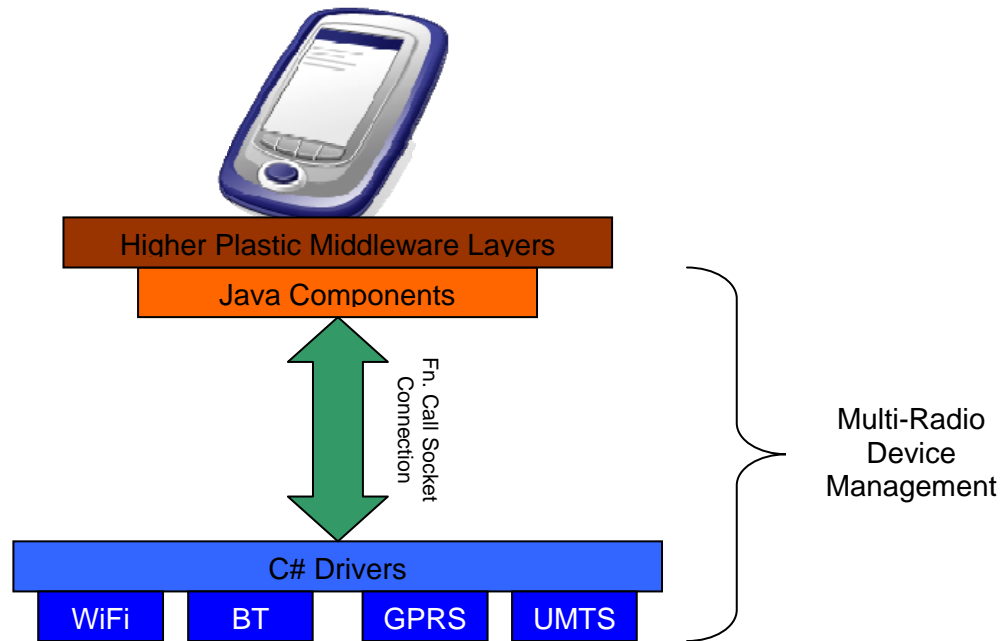


Figure 2. Multi-Radio Device Management implementation using a local socket connection

Development status

Version	What's available
1.0.0	Java class representation of network radio adapters. Local socket communication and protocol implementations for calling C# based drivers in Java class representation of network radio adapters. C# based drivers for communication with WLAN, GPRS/UMTS, WidComm Bluetooth stack (fully) and Microsoft Bluetooth stack (partially due to limitation of hardware support) radio adapters on Windows Mobile 2003, 5 and 6 based PDAs.

More specifically, the current C# implementation status of the three supported radio network adapters is detailed as follows:

The status of WiFi implementation:

1. Supports IEEE 802.11 a/b/g networks.
2. Supports switch on and off of the WiFi network adapter on a Windows Mobile PDA.
3. Supports the discovery of WiFi networks in the surrounding environment of a Windows Mobile PDA, which includes both infrastructure-based (e.g., Access Points) and infrastructure-less WiFi networks (e.g., ad-hoc WiFi networks). Information related to the discovered WiFi networks include: network name, its MAC address, network mode (i.e., infrastructure-based or infrastructure-less), network encryption mode and network signal strength.

4. Supports the connection and disconnection to both infrastructure-based and infrastructure-less networks. Note: for networks with hidden SSIDs or WAP/WEF encryptions, it is necessary to pre-configure these networks through the WiFi manager on a Windows Mobile PDA before being able to connect.
5. Supports the retrieval of the IP address of a WiFi connection after a Windows Mobile PDA has successfully connected to a WiFi network.
6. Supports the retrieval of the MAC address of the WiFi adapter on a Windows Mobile PDA, assuming that it has already been switched on successfully.
7. Miscellaneous notes:
 - a. To be able to successfully connect to an ad-hoc WiFi network, the WiFi adapter on the connecting device needs to be switched on before switching on the WiFi adapter on the hosting device.

The status of cellular network implementation (i.e., GPRS, EDGE and UMTS):

1. Supports the connection and disconnection to a cellular network, providing that the network has been configured correctly on a Windows Mobile PDA.
2. Supports the retrieval of the IP address of a cellular connection after a Windows Mobile based PDA has successfully connected to a cellular network.
3. Supports the retrieval of the IMEI code of the installed SIM card on a Windows Mobile based PDA.
4. Miscellaneous notes:
 - a. To be able to successfully use the above functions, you need to have authorized access to a SIM card (e.g., have already entered your SIM Pin number).
 - b. With the current implementation it is not possible to switch on and off the cellular network interface on a Windows Mobile based PDA due to unavailability of proper library for Windows Mobile.
 - c. It is possible to connect to the following cellular networks: GPRS, EDGE, UMTS and HSDPA. However, a network provider might decide which type of cellular network a PDA should connect to, based on the availability of cellular stations in the surrounding environment. Therefore, it is not possible to specify which type of cellular network to connect to and this could change frequently during a connection.
 - d. If a PDA has already connected to the Internet using a WiFi Access Point, then you must disconnect from the WiFi AP before being able to connect to a cellular network. In contrary, if you have already connected to a cellular network, you don't need to disconnect from the network to connect to a WiFi AP as it automatically overrides the cellular connection.
 - e. A Windows Mobile PDA can connect to a cellular network and other ad-hoc networks (i.e., WiFi or Bluetooth) at the same time.

The Bluetooth implementation is further categorized into two types based on the underlying Bluetooth stack installed on a Windows Mobile based PDA (i.e., WidComm Bluetooth stack and Microsoft Bluetooth stack).

The status of Bluetooth implementation is as follows:

1. WidComm Bluetooth stack:

- a. Supports switch on and off of the Bluetooth network adapter on a Windows Mobile based PDA.
- b. Supports the discovery of Bluetooth networks in the surrounding environment. Information related to the discovered Bluetooth networks include: network name, its MAC address and network signal strength.
- c. Supports the connection and disconnection to the following Bluetooth networks: PAN, NAP and LAP.
- d. Supports the retrieval of the IP address of a BT connection after a Windows Mobile PDA has successfully connected to a BT network.
- e. Supports the retrieval of the address of a Bluetooth adapter on a Windows Mobile PDA, assuming that it has been switched on successfully.
- f. Miscellaneous notes:
 - i. To be able to successfully connect to a Bluetooth network, it assumes that the hosting device offers the connecting Bluetooth network and has been configured correctly.
 - ii. If the code complains about the trial license of Franson BlueTools is expired, then please go to the web site <http://franson.com/bluetools> to obtain the new license code. Once you have obtained the code, replace the old code on line 94 of WidcommBluetoothDevice.cs with the new one (i.e., `license.LicenseKey = "HU5Uafp132NNNWB8oiigVkaQfu10ORLbHY7Z";`).

2. Microsoft Bluetooth stack:

- a. Supports switch on and off of the Bluetooth network adapter on a Windows Mobile PDA.
- b. Supports the discovery of Bluetooth networks in the surrounding environment. Information related to the discovered Bluetooth networks include: network name, its MAC address and network signal strength.
- c. Supports the connection and disconnection to the following Bluetooth networks: NAP and LAP.
- d. Supports the retrieval of the IP address of a BT connection after a Windows Mobile PDA has successfully connected to a BT network.
- e. Supports the retrieval of the Bluetooth address of a Bluetooth adapter on a Windows Mobile PDA, assuming that it has been switched on successfully.
- f. Miscellaneous notes:
 - i. To be able to successfully connect to a Bluetooth network, it assumes that the hosting device offers the connecting Bluetooth network and has been configured correctly.
 - ii. The implementation supports the connection to a PAN based network; however, there is no Microsoft Bluetooth stack based PDA currently on the market which supports the PAN profile.

Intended audience

Users who would like to test the supported functionalities of the Multi-Radio Device Management layer.

License

Copyright (C) 2006-2008 PLASTIC CONSORTIUM¹

The PLASTIC Multi-radio Networking library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details².

Language

Java

C#

Environment (set-up) info needed if you want to run this sw (service)

The deployment and testing of the Multi-Radio Device Management layer require two devices: a workstation for deploying the Multi-radio Device Management layer software modules and a Windows Mobile PDA for testing. The specifications of the hardware and software components of the devices are detailed in Section 2.1 System requirements.

Platform

Windows XP for deployment

Windows Mobile 2003, 5 or 6 for testing

Documents

Design	PLASTIC Deliverable 3.1
Prototype implementation	PLASTIC Deliverable 3.2
Assessment and revision	PLASTIC Deliverable 3.3
Developer's guide	Middleware Developer's Guide: Multi-radio Device Management Layer

¹ <http://www.ist-plastic.org>

² <http://www.gnu.org/licenses/lgpl.html>

2 Deployment

2.1 System requirements

The requirements are for two devices: a workstation for deploying the Multi-radio Device Management layer software modules and a Windows Mobile PDA for testing.

Workstation requirements:

- Hardware requirements (minimum): 600MHz processor, 192MB RAM and 3GB hard disk space. Recommended: 3.4GHz processor, 2GB RAM and 5GB hard disk space
- Windows XP with Service Pack 2
- Microsoft Visual Studio 2005 Professional Edition with Compact Framework 2.0
- Eclipse IDE version 3.2 or later
- Fat Jar Eclipse Plug-in (<http://fjep.sourceforge.net/>)
- Microsoft ActiveSync 4.5 or later
- J2SE 1.4.2 SDK
- Windows Mobile 6 Professional SDK for deployment to Windows Mobile 6 devices (optional)

Windows Mobile PDA requirements:

- Hardware requirements (minimum): 168MHz processor, 64MB ROM and 64MB RAM. Recommended: 400MHz processor, 128MB ROM and 64MB RAM.
- Supports multiple network connectivity: WiFi, cellular networks (i.e., GPRS, EDGE and UMTS) and Bluetooth. Note: it is possible to use the software module with only one radio network adapter, however for multi-radio based testing it is recommended that the testing PDA has at least two different radio network adapters.
- Windows Mobile 2003 for Pocket PC or later. Windows Mobile 6 Professional Edition is strongly recommended for optimal network behavior. Note: the current software module does not support Windows Mobile for Smartphones (i.e., phones without touch screens).
- Mysaifu JVM for PocketPC (http://www2s.biglobe.ne.jp/~dat/java/project/jvm/index_en.html).
- Magic Button (<http://www.trancreative.com/mb.aspx>), a task manager for Windows Mobile PDAs.
- The following devices have been tested for the implementation: iPAQ h6340, HTC TyTN and HTC Kaiser.

2.2 Download

The Multi-radio Networking package can be downloaded from the following URL:

<http://www-c.inria.fr/plastic/workpackage/wp3/development/plastic-multiradio-communication-v1.zip/view>.

2.3 Configure

Simply load the Java code and the C# code in Eclipse and Microsoft Visual Studio 2005, respectively.

2.4 Compile and deploy

To test the MultiRadioDeviceManagement module on a Windows Mobile PDA, you need to deploy both the C# based drivers as well as the Java classes to the device. Once both parts are deployed to the PDA, the C# based drivers are called automatically when the Java part starts running.

The C# based driver module can be compiled and deployed as follows:

1. Connect the PDA to your workstation using ActiveSync.
2. Compile and deploy the MultiRadioDeviceManagement project to your PDA using Microsoft Visual Studio 2005. By default, it deploys an executable named csharplistener.exe to your root directory. This is necessary for the Java code to communicate with the C# code. Also, it is important that you specify the correct mobile device platform for the deployment (i.e., Windows Mobile 5, 6 or PocketPC 2003).
3. Copy the following library files from the /libs subdirectory to the root directory of the PDA or where csharplistener.exe is located: BlueTools.dll, BlueToolsMS.dll, BlueToolsWC150.dll, BlueToolsWC.dll, BTAccess.dll, Franson.BlueTools.200.CF.dll, and InTheHand.Net.Bluetooth.dll.

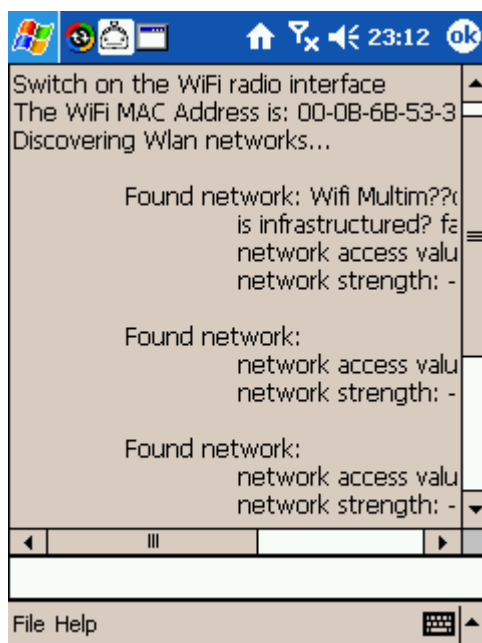
The Java based classes of the Multi-radio Device Management module can be compiled and deployed as follows:

1. Export all the classes and reference libraries into a single jar file using the Fat Jar Eclipse plug-in. During export, specify the Main class that will be executed in the jar file; this is used by the Mysaifu JVM to determine how to run the Java module.
2. Connect the PDA to the desktop using ActiveSync and copy the exported jar file to the PDA.
3. Start Mysaifu JVM and specify the jar file as the running type, configure related options (e.g., memory) and start the execution. When the Java code starts running, it automatically calls the C# driver within the code. Note: you can now start Magic Button to switch between the Java code and the C# code for debugging the program.

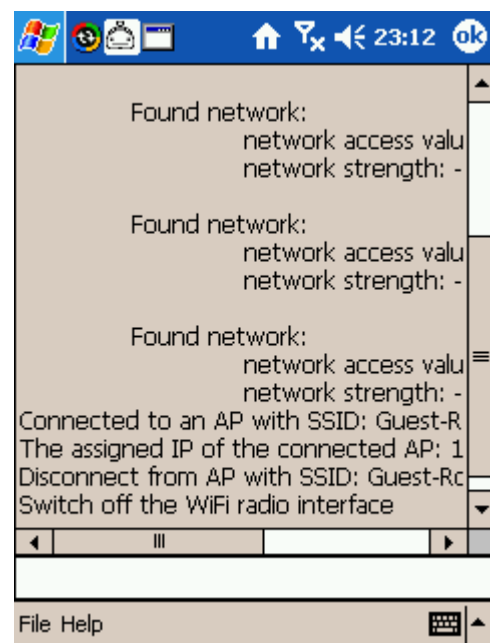
3 Tutorial

The Multi-Radio Device Management layer should be utilized by upper middleware layers or applications. However, it is possible to test the Multi-Radio Device Management layer individually to verify the underlying radio network interfaces. To test the module independently, you need to specify `org.plastic.network.model.ipaq_impl.Test` as the Main class when running the module under Mysaifu. The Test class uses a single argument to determine which type of test it should run; the argument currently supports the following values: 'wifi', 'bt', 'gprs' and 'all' (e.g., use "Test wifi" to test the WiFi adapter and "Test all" to test all adapters).

The following screenshots show the independent testing of WiFi, Bluetooth and GPRS adapters on an iPAQ h6340. The screenshots include both Java and C# outputs:



WiFi Java output



WiFi Java output cont...

```

Received: SWITCH_ON_WIFI
Sent: ACK_SUCCEED
Received: GET_WIFI_MAC
Sent: 00-0B-6B-53-30-D4
Received: DISCOVER_WIFI_NETWORKS
Sent: +Wifi Multim??dia+l+True+-9
received connect command WiFi is
Parameter 0 is: Guest-Rocq
Parameter 1 is: true
Parameter 2 is:
Sent: ACK_SUCCEED
Received: GET_WIFI_IP
Sent: 128.93.135.244
Received: DISCONNECT_WIFI
Sent: DISCONNECT_WIFI
Received: SWITCH_OFF_WIFI
Sent: ACK_SUCCEED

```

WiFi C# output

```

Switch on GPRS
Connect to the GPRS network? true
The assigned IP of the GPRS connection
The IMEI of the SIM card is: 352795002
Switch off GPRS

```

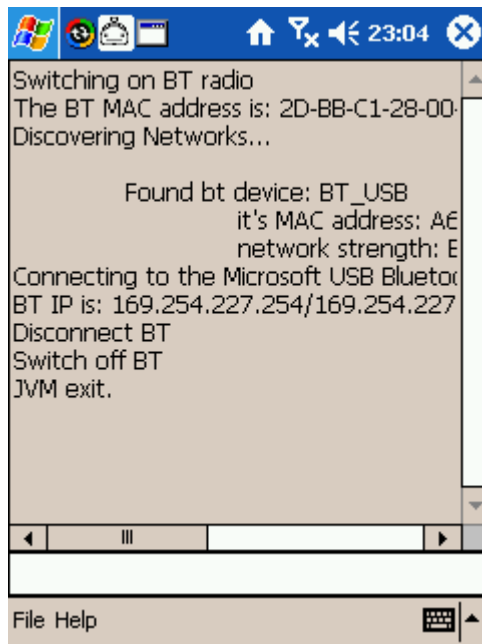
GPRS Java output

```

Received: SWITCH_ON_GPRS
Sent: ACK_SUCCEED
Received: GET_GPRS_IP
Sent: 352795002189776
Received: SWITCH_ON_GPRS
Sent: ACK_SUCCEED
Received: CONNECT_GPRS
Sent: ACK_SUCCEED
Received: GET_GPRS_IP
Sent: 10.2.7.250
Received: DISCONNECT_GPRS
Sent: ACK_SUCCEED
Received: SWITCH_OFF_GPRS
Sent: ACK_SUCCEED

```

GPRS C# output

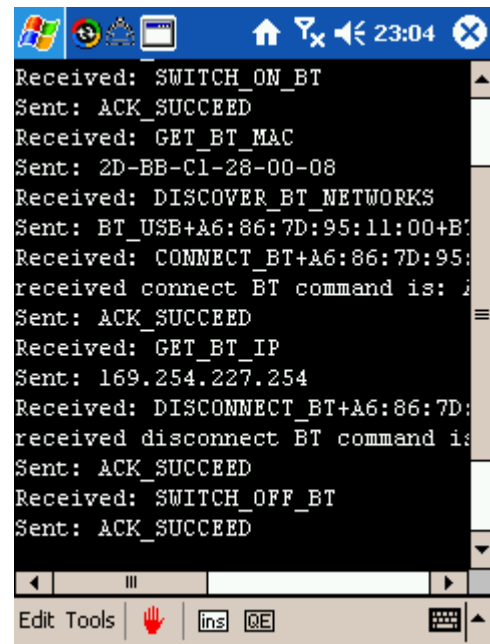


A screenshot of a Java application window. The title bar shows standard Windows icons and the time 23:04. The main text area contains the following output:

```
Switching on BT radio
The BT MAC address is: 2D-BB-C1-28-00-
Discovering Networks...

    Found bt device: BT_USB
                    it's MAC address: A6:86:7D:95:11:00
                    network strength: E
Connecting to the Microsoft USB Bluetooth
BT IP is: 169.254.227.254/169.254.227
Disconnect BT
Switch off BT
JVM exit.
```

The window has a menu bar with 'File' and 'Help'.

Bluetooth Java output

A screenshot of a C# application window. The title bar shows standard Windows icons and the time 23:04. The main text area contains the following output:

```
Received: SWITCH_ON_BT
Sent: ACK_SUCCEED
Received: GET_BT_MAC
Sent: 2D-BB-C1-28-00-08
Received: DISCOVER_BT_NETWORKS
Sent: BT_USB+A6:86:7D:95:11:00+B7
Received: CONNECT_BT+A6:86:7D:95:
received connect BT command is: 2
Sent: ACK_SUCCEED
Received: GET_BT_IP
Sent: 169.254.227.254
Received: DISCONNECT_BT+A6:86:7D:
received disconnect BT command is:
Sent: ACK_SUCCEED
Received: SWITCH_OFF_BT
Sent: ACK_SUCCEED
```

The window has a menu bar with 'Edit' and 'Tools'.

Bluetooth C# output

4 References

- [PCD1.2] Deliverable D1.2. "Formal description of the PLASTIC conceptual model and of its relationship with the PLASTIC platform toolset".
- [PCD3.1] PLASTIC Consortium. "Deliverable D3.1 - Middleware Specification and Architecture". March 2007.
- [PCD3.2] PLASTIC Consortium. "Deliverable D3.2 - Middleware prototype implementation". August 2007.
- [PCD3.3] PLASTIC Consortium. "Deliverable D3.3 - Middleware: Assessment and Revision".